



freeBSDTM **JOURNAL**TM
Jan/Feb 2017

Globally Accessible FreeBSD

FlightAware
and FreeBSD

Improving
FreeBSD

**TRANSLATION
TOOLS**

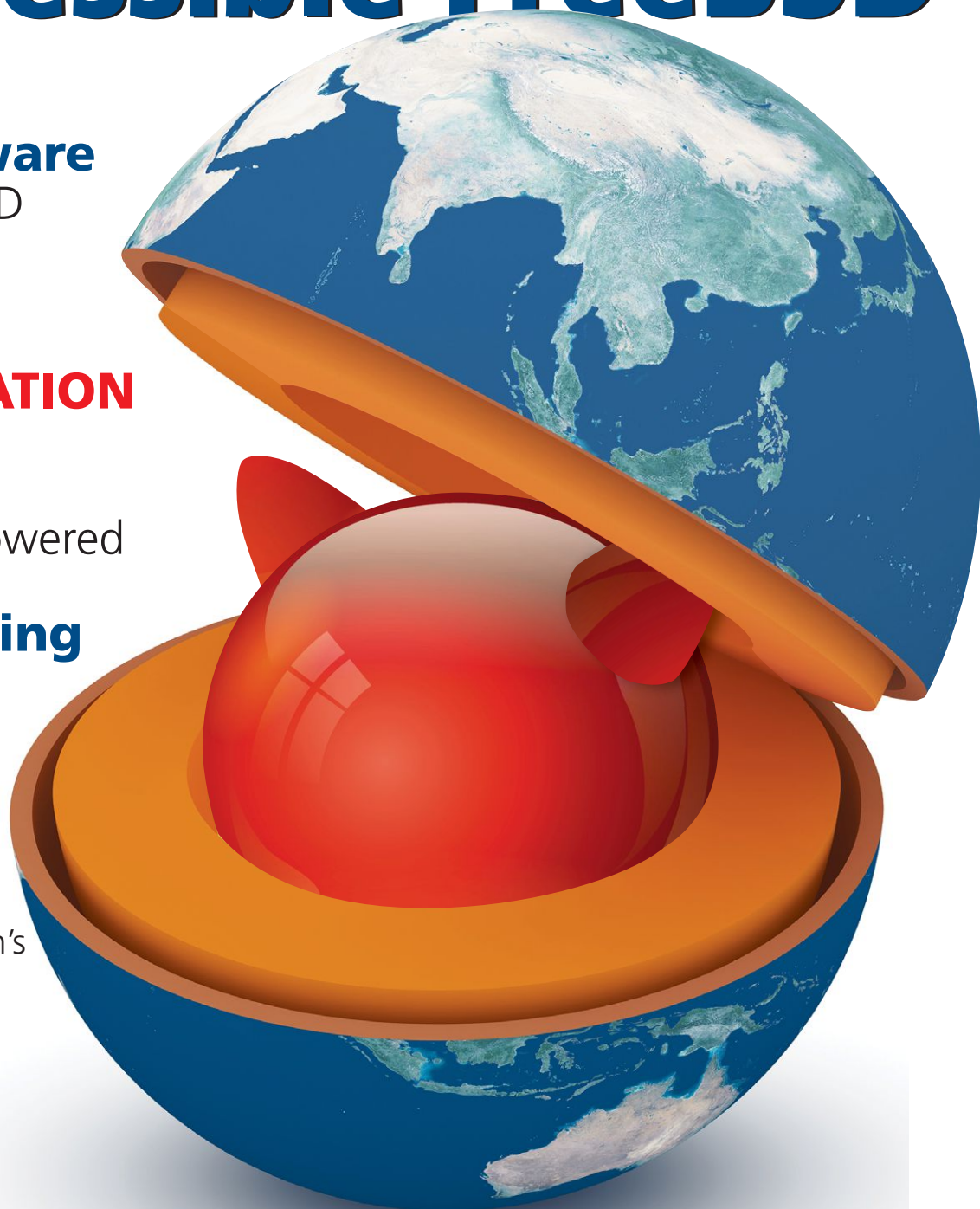
FreeBSD-powered

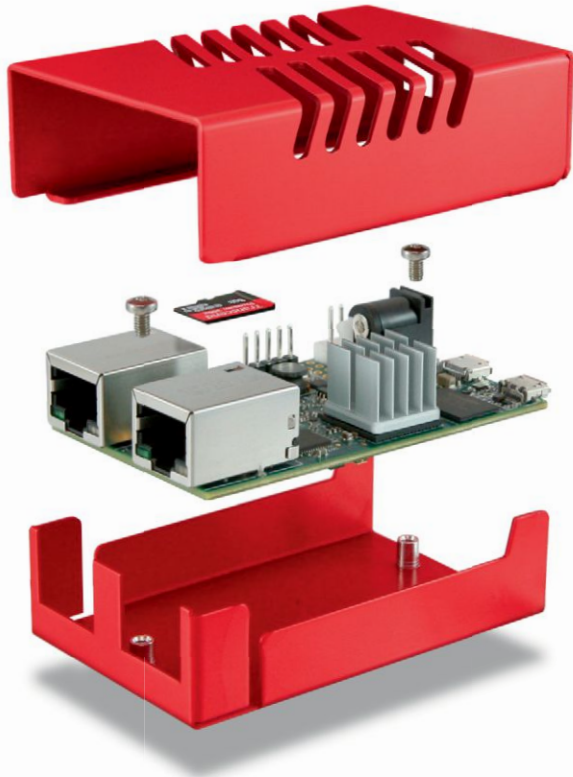
**LCD
Advertising
in China**

Also:

The Foundation's

**GLOBAL
EFFORTS**





INTRODUCING THE **SG-1000** microFirewall

Shipping Now!

\$149

Includes pfSense® Gold, a \$99 value.



You asked. We delivered! The new Netgate® SG-1000 microFirewall is a cost-effective, state-of-the-art, ARM®-based, pfSense Security Gateway appliance. The SG-1000 comes with dual 1Gbps Ethernet ports, enabling maximum throughput exceeding 300 Mbps. The ARM Cortex®-A8 in the TI AM3552 SoC and DDR3L RAM combine to facilitate low power consumption while maintaining performance. The SG-1000 comes in a lightweight and durable anodized aluminum case. Its credit-card sized form-factor allows it to be easily tucked away, but you'll be proud to show it off.

- The ideal security endpoint for the Internet of Things (IoT)
- SCADA firewall for infrastructure & commercial systems
- Secure multi-dwelling units (MDU) such as dorm rooms & apartments
- Deploy as an IPMI port firewall to close critical IPMI / BMC vulnerabilities
- Portable VPN endpoint firewall, network tap or on-premises SMB appliance
- 5 V DC power for a wide range of applications with low energy consumption



Shop now at the Netgate store or authorized partners worldwide.

www.netgate.com/products/sg-1000.html

pfSense® is a registered trademark of Electric Sheep Fencing, LLC. Netgate is a registered trademark of Rubicon Communications, LLC.
Other names and brands may be claimed as the property of others..



Table of Contents

FreeBSDTM JOURNAL

January/February 2017

3 Foundation Letter

Welcome to FreeBSD Journal 2017!

By George Neville-Neil

24 New Faces

In this installment, the spotlight is on Nikolai Lifanov (ports) and Konrad Witaszczyk (src), who became committers in November 2016, and Larry Rosenman and Jean-Sébastien Pédron, who received ports commit bits in January 2017.

By Dru Lavigne

28 Book Review

Aditya Y. Bhargava's *Grokking*

Algorithms: An Illustrated Guide for Programmers and Other Curious

People. The goal of this book is to be an easy on-ramp to learning more about algorithms and to help dispel the myth that algorithms are an obscure and complex subject. *By Steven Kreuzer*

30 svn Update While activity in base tapered off a bit in December, the author did see a few commits that he describes as "interesting."

By Steven Kreuzer

32 Events Calendar

By Dru Lavigne



INTERACTING

with the FreeBSD Project

The Foundation's Global Efforts

22 The FreeBSD Foundation is proud to serve a welcoming international community and to support and grow FreeBSD awareness throughout the world. *By Anne Dickison*

Globally Accessible FreeBSD

FlightAware and FreeBSD

It is the greatest, mostly. Up until very recently, FreeBSD was used almost entirely to power the site and services with only a handful of exceptions. This, however, is starting to change and the author explains why. *By Sean Kelly*

4

FreeBSD-powered LCD Advertising Displays (in Some Waiting Rooms Along the High-speed Railway of China).

In 2011 and 2013, a number of freestanding LCD advertising displays were installed in waiting rooms along some of China's high-speed railways. In this article, the author covers the business use case—excluding the network server, the user experience on stability, and the pain points on the GPU. *By Li, Xiao*



10

Improving the FreeBSD Translation Tools.

A large portion of the world is inhabited by people who could benefit from using FreeBSD—and can't—because they don't speak English even as a second or third language. There is a huge benefit that we can bring to them.

By Warren Block

16



IS AFFORDABLE FLASH STORAGE OUT OF REACH?

NOT ANYMORE!

IXSYSTEMS DELIVERS A FLASH ARRAY FOR UNDER \$10,000

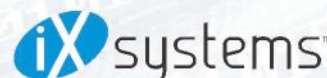
Introducing FreeNAS® Certified Flash. A high performance all-flash array at the cost of spinning disk.

KEY ADVANTAGES

- ⚡ 10TB of all-flash storage for less than \$10,000
- ⚡ Unifies SAN/NAS for block and file workloads
- ⚡ Runs FreeNAS, the world's #1 software-defined storage solution
- ⚡ OpenZFS ensures data integrity
- ⚡ Scales to 100TB in 2U
- ⚡ Perfectly suited for Virtualization, Databases, Analytics, HPC, and M&E
- ⚡ Performance-oriented design provides maximum throughput/IOPs and lowest latency
- ⚡ Maximizes ROI via high-density SSD technology and inline data reduction

The all-flash datacenter is now within reach. Deploy a FreeNAS Certified Flash array today from iXsystems and take advantage of all the benefits flash delivers.

For more information, visit iXsystems.com/FreeNAS-Certified-Servers today.



- John Baldwin • Member of the FreeBSD Core Team and Co-Chair of *FreeBSD Journal* Editorial Board
- Brooks Davis • Senior Software Engineer at SRI International, Visiting Industrial Fellow at University of Cambridge, and past member of the FreeBSD Core Team
- Bryan Drewery • Senior Software Engineer at EMC Isilon, member of FreeBSD Portmgr Team, and FreeBSD Committer
- Justin Gibbs • Founder and President of the FreeBSD Foundation and a Senior Software Architect at Spectra Logic Corporation
- Daichi Goto • Director at BSD Consulting Inc. (Tokyo)
- Joseph Kong • Senior Software Engineer at EMC and author of *FreeBSD Device Drivers*
- Steven Kreuzer • Member of the FreeBSD Ports Team
- Dru Lavigne • Director of the FreeBSD Foundation, Chair of the BSD Certification Group, and author of *BSD Hacks*
- Michael W. Lucas • Author of *Absolute FreeBSD*
- Ed Maste • Director of Project Development, FreeBSD Foundation
- Kirk McKusick • Director of the FreeBSD Foundation and lead author of *The Design and Implementation* book series
- George V. Neville-Neil • Director of the FreeBSD Foundation, Chair of *FreeBSD Journal* Editorial Board, and coauthor of *The Design and Implementation of the FreeBSD Operating System*
- Hiroki Sato • Director of the FreeBSD Foundation, Chair of Asia BSDCon, member of the FreeBSD Core Team, and Assistant Professor at Tokyo Institute of Technology
- Benedict Reuschling • Vice President of the FreeBSD Foundation and a FreeBSD Documentation Committer
- Robert Watson • Director of the FreeBSD Foundation, Founder of the TrustedBSD Project, and University Senior Lecturer at the University of Cambridge

S&W PUBLISHING LLC
PO BOX 408, BELFAST, MAINE 04915

- Publisher** • Walter Andrzejewski
walter@freebsdjournal.com
- Editor-at-Large** • James Maurer
jmaurer@freebsdjournal.com
- Copy Editor** • Annaliese Jakimides
- Art Director** • Dianne M. Kischitz
dianne@freebsdjournal.com
- Office Administrator** • Michael Davis
davism@freebsdjournal.com
- Advertising Sales** • Walter Andrzejewski
walter@freebsdjournal.com
Call 888/290-9469

FreeBSD Journal (ISBN: 978-0-615-88479-0) is published 6 times a year (January/February, March/April, May/June, July/August, September/October, November/December).
Published by the FreeBSD Foundation,
5757 Central Ave., Suite 201, Boulder, CO 80301
ph: 720/207-5142 • fax: 720/222-2350
email: info@freebsdjournal.org
Copyright © 2017 by FreeBSD Foundation. All rights reserved.

This magazine may not be reproduced in whole or in part without written permission from the publisher.

Welcome to *FreeBSDTM Journal* 2017!

The Editorial Board is excited about the selection of topics planned for our 2017 issues and, of course, the articles in this, the first, issue of the new year. First up is Sean Kelly's article on how FlightAware uses FreeBSD to provide accurate flight data. If you travel a lot, then you probably know about flightaware.com and their free app. When you use it to find out why you're delayed, yet again, you're using FreeBSD. Knowing that you're using FreeBSD may take a bit of the sting out of that two-hour ground delay!

This issue's other two feature articles relate to FreeBSD's use in various languages. Li, Xiao writes about FreeBSD and railway information signs in China, and Warren Block discusses the new slate of translation tools in use by the FreeBSD documentation and development teams. The international focus is rounded out by a piece from Anne Dickison of the FreeBSD Foundation, outlining how the Foundation is working to make sure FreeBSD is known and loved around the world.

As we're just beginning the year, we do not have a conference report for this issue, but in later issues we'll report on FOSDEM 2017, which just finished a couple of weeks ago, as well as AsiaBSDCon (<https://2017.asiabsdcon.org>) planned for Tokyo, Japan, in March; BSDCan (<http://www.bsdcan.org/2017/>) in Ottawa, Canada, in June; as well as BSDCam, VBSDCon, and EuroBSD, taking place in the months of August and September.

In this issue, you'll enjoy columns from Dru Lavigne, who profiles New Faces in the FreeBSD community, and Steven Kreuzer, who reviews *Grokking Algorithms: An illustrated guide for programmers and other curious people*, by Aditya Y. Bhargava, and also delivers a new installment of his svn update column.

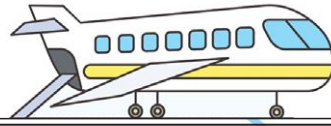
2017 is shaping up to be a great year for the *FreeBSD Journal*. We are working on issues on security, configuration management, and resource control for the first half of this year. And for now, we'll just keep the rest of the year under our horns.

Hope to see everyone at a BSD event in 2017!

George Neville-Neil

Editor in Chief of the *FreeBSD Journal*

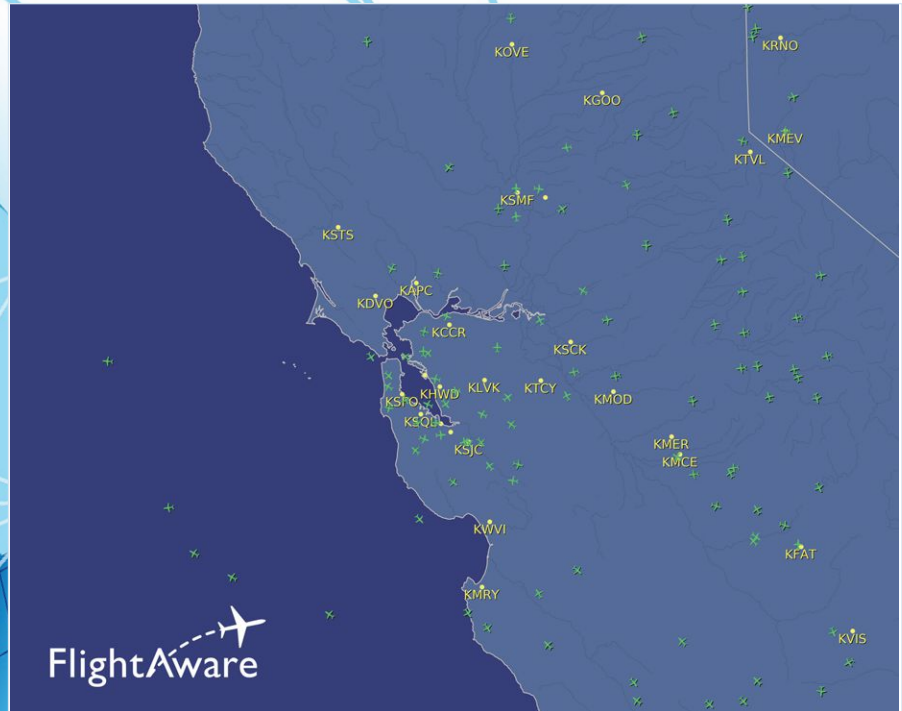
Director of the FreeBSD Foundation



By Sean Kelly

FlightAware and *FreeBSD*

It is the greatest, mostly.



If you've not heard of us, FlightAware is the world's largest flight-tracking data company, providing over 10,000 aircraft operators and service companies with global flight-tracking solutions. We provide free flight tracking to the general public on our website and via our mobile apps, but we also have other commercial products and services used by aircraft operators around the world. We accomplish this by aggregating data from our own ground stations, government data sources covering over 55 countries, and other satellite-based providers. We then take all this data, interpret it, and turn it into more holistic views of what is going on. As you might expect, this requires a fair amount of processing and storage.

At FlightAware, our business to date has been built on top of FreeBSD, and we are very strong proponents of it. We have four employees who are inactive members of the FreeBSD Project, including our CEO and CTO. Up until very recently, FreeBSD was used almost entirely to power our site and services with only a handful of exceptions. This, however, is starting to change and I'd like to explain why. But first, let's cover our history with FreeBSD and how we've been using it.

Our CEO used to run ftp2.freebsd.org and he contributed to the ports tree. He has been using FreeBSD since the 2.0-ALPHA days and continues using it both professionally and personally to this day. Our CTO goes further back, to the beginnings of 386BSD, where he maintained patch kits and later contributed loopback detection to the T1 driver. Going back even further, one of our developers did a lot of hacking on 2BSD and 4.1BSD and later helped with the *FreeBSD Handbook*. Finally, I started using FreeBSD at the release of 4.0 and have contributed some patches as well as the initial implementation of the software watchdog subsystem.

Along with FreeBSD, we use the Tcl programming language extensively. This includes our website, which is written in Tcl and served by Apache's `mod_rivet` extension. Several ports of our Tcl packages exist in the ports tree, including `speedtables`, `casstcl`, `yajl-tcl`, `tclauncher`, `tcldreadline`, and `tcldbsd`. These ports are maintained by `tcltk@` and Pietro Cerruti. While Pietro is not a FlightAware employee, he generously updates ports to coincide with our software releases.

Some of the reasons that we do use FreeBSD include the stability and robustness of the network stack, the cohesive community, the holistic view of the system from kernel to userland, and ZFS. All these things combine to create an optimum and trustworthy open-source environment for our servers and services.

How We Use It

Like many others, we run web servers, Varnish servers, mail servers, and DNS servers on FreeBSD. While not entirely unusual, there are several other things we do on the FreeBSD platform that may interest others a bit more. These include a custom FreeBSD installer, high transaction RDBMS, a few different in-memory databases using shared memory, and, of course, all of these utilize the fantastic ZFS filesystem.

Installation: A Server Is Born

While `bsdinstall` is a great improvement on `sysinstall`, we still needed something that provided much more automation. Almost all of our servers start with the same base install, so it was unnecessary for us to repeatedly punch the same information into `bsdinstall`. Instead, we developed a set of scripts that incorporate `bsdinstall` and `bsdconfig`. These scripts run from a `mfsBSD` PXE environment and do the following:

- Destroy any existing hardware RAID configuration and partitions
- Wipe any existing ZFS labels on disks
- Create a new `zpool` on all internal disks
- Create all the appropriate ZFS filesystems to support boot environments
- Pull down the FreeBSD installer files and untar them
- Create a user
- Configure all network parameters

Once that is complete, the machine reboots and we're prompted with a menu where we can select the components we'd like installed. Each component is managed by a shell function inside of a shell script, making it easy to manage and modify. When we need a new component, we just create a new function.

All this evolved over time. Ideally, we'd like to be able to have an installer that will do a PXE boot, find a configuration file matching a MAC address, and do a full headless installation from that configuration file. It would be even better if this was built into FreeBSD so that FreeBSD had an easy zero-touch deployment system that was standardized and each shop didn't have to roll their own.

Traditional Databases: Make It Fast

While it is less novel than our custom installer, another use for FreeBSD at FlightAware is to power our PostgreSQL database cluster. We chose FreeBSD for this especially due to the data integrity features provided by ZFS. Another nice benefit is the LZ4 compression which yields a compression ratio of around 2.15x with a negligible performance impact. The cluster itself is a standard PostgreSQL 9.4 setup using the native streaming replication support. We use `pgpool` to load balance read-only transactions while all write transactions go to the master. This allows us to spread queries out over a larger number of servers,





FlightAware and FreeBSD

increasing our transaction processing potential due to most transactions being read-only.

In October 2015, we transitioned our PostgreSQL servers away from 24 hard disks in ZFS mirror pairs to use four NVMe SSDs in mirrored pairs. This was a huge performance gain once we got past issues with TRIM that eventually resulted in us disabling it. The NVMe subsystem timed out upon zpool create since ZFS was trying to TRIM the entire pool and NVMe gave up waiting on the SSD to do it. We also had issues during normal operation when TRIM was executed on Samsung SSDs that caused several outages. When PostgreSQL would clean up `pg_xlog` files, it would essentially briefly stall I/O due to TRIM. We've since switched to Intel SSDs which don't seem to exhibit the problem, but we left TRIM off anyway.

To back up all of this, we developed a custom toolset in Tcl that manages ZFS snapshots. The tool takes a snapshot on a database server, sends it to another host where it is staged, and from there the snapshot is dispersed to various sites. These same tools also manage the retention policy for the snapshots, keeping a certain amount of hourly, daily, and weekly snapshots on hand.

In-Memory Databases: Making Fast Faster

Along with our PostgreSQL databases, we also have internal software called Birdseye. Birdseye is an in-memory database written in Tcl and C, and based on our open-source speedtables project. Birdseye uses shared memory to store approximately 24 hours of positions for all aircraft that we know about. Multiple Birdseye processes on a single machine share the same shared memory segments so that more clients can be serviced concurrently. Clients, such as the website, can then connect to these Birdseye servers and make all sorts of queries to very quickly discern what a plane is doing, has done, or even what is happening within an arbitrary geographical box.

Along with Birdseye, we have another in-memory database of sorts called Superbird. Superbird, too, is based on our speedtables project and is written in Tcl with some generated C. Superbird will periodically read in any changes that have been made to PostgreSQL tables and keep them in an in-memory speedtables database. This is leveraged by the website so that it is possible to make faster queries of busy tables by accessing a local in-memory cached copy rather than needing to access an off-host PostgreSQL server. In other words, Superbird is a database caching layer utilizing update polling.

As you can see, shared memory is an important technology for us. We use it quite extensively with our internal tools as well as with third party projects like PostgreSQL. We've found that the overall performance of SHM has gotten better in the last several years and hope that it continues to do so. We noticed significant improvements with FreeBSD 9 and a bit again with FreeBSD 10.

Pain Points: Not Everything Is Perfect

Now that we've covered how FreeBSD works for us, we'd be remiss to not discuss our pain points. FlightAware is rapidly growing, as is our need to be able to deploy new systems and services at an ever-increasing rate. Further, data growth and increasing ingestion rates are causing us to reevaluate how we store, process, and analyze our datasets. This is where our choice of FreeBSD becomes harder to continue and justify.

Installation: An Arm of Servers

One amazing thing about FreeBSD is that it can be installed on a system and then mostly forgotten. You obviously want to install updates and so forth, but the system is generally rock solid and requires very little ongoing maintenance and care. As a result, we tend to treat our FreeBSD systems as unique rather than as a fleet of entirely dispensable resources. We'll add and remove software, but rarely do we find the need to do a clean install. The problem with this is that the approach doesn't scale well. As a result, we're starting to treat operating system installs as a commodity.

As I discussed earlier, we've built our own FreeBSD installer that mostly works for us to address this change. It isn't great and we'd like to add more features, such as the ability for it to TFTP a configuration file for entirely headless installs. Maybe there would be a base configuration file and then there could be a MAC address-based or SMBIOS system serial number-based override file. But instead of us implementing this for our own tools, it'd be fantastic if this was just part of the FreeBSD installer.

The FreeBSD installer should support entirely headless installations with documented scripting functionality. This is something that the Linux world conquered long ago with the Red Hat kickstart system or the Debian preconfiguration file. The installer could be PXE booted and then use DHCP options to direct it to where it can download configuration data. This would make it trivial to deploy or redeploy hundreds of servers at once.

Containers and Isolation: When Jail Isn't Enough

Now that I've shared my vision for how the FreeBSD installation process could be improved, let me explain what it is we want to do with fleets of heedlessly installed servers.

We are rapidly moving away from deploying services on a standard system installation. Like a lot of cloud scale companies, we are adopting a containerization approach that will allow us to easily deploy many services on a single piece of hardware without the software and library dependencies that this traditionally introduces. We'd like to have every base installation be identical and serve as nothing but a platform for hosting application containers. This simplifies installation and management.

As one potential path toward this, we did some evaluation of jails and found them mostly serviceable. However, we opted not to move in that direction due to several issues we faced. One glaring issue is the lack of per-jail shared memory. FreeBSD jails' shared memory is not isolated from each other or from the base system. The lack of separation means services like PostgreSQL and Birdseye can't be securely deployed in adjacent jails without special care to avoid conflict or exploitation. For example, running PostgreSQL in adjacent jails is risky due to the pgsq user having the same UID and thus each jail having access to the other jail's IPC resources.

Despite all of this, a lot of progress seems to have been made by adding RCTL to GENERIC, but we'd also like to see more focus in this area. VIMAGE should be made stable and added, as well. Further, we'd like to see more namespacing for things like shared memory implemented to provide true isolation. This is one area where Linux is ahead and has resulted in technologies like Docker and Kubernetes thriving.

The actual container mechanism is important, but how it is managed is also a big deal. As it is, FreeBSD has many ways to manage jails that are in various states of support and development. We've got ezjail, iocage, CBSD, jail(8)'s /etc/jail.conf, and many other homegrown tools and derivatives. While choice is almost always a good thing, this diverse ecosystem also makes it harder for automation and orchestration tools to target support for jails. It would be fantastic if there was a way as part of base to bring all of the diverse features under one roof—one set of APIs and commands to rule them all.



64-bit Java: Twice as Good as Before

As I already mentioned, our datasets are growing as is the amount of data we ingest. This has caused us to turn to newer technologies like Cassandra, Kafka, and Spark to store, move, and make sense of it all. All these have one thing in common: they run on the Java Runtime Environment.

There are not a lot of options for Java on FreeBSD. You can build and use the OpenJDK open-source implementation of Java which can be compiled into a native 64-bit implementation. You can also use the 32-bit Linux JRE from Oracle through the FreeBSD Linux emulation. However, only until very recently have 64-bit Linux system calls been supported and we've not successfully gotten the 64-bit Oracle JRE to run on FreeBSD. Finally, The FreeBSD Foundation used to offer a FreeBSD native binary for the Oracle JRE, but that is no longer available.

So you're probably wondering what our problem is and why we don't just use OpenJDK. The answer is that we've encountered issues with it that were hard to pin down. With Kafka, we saw periodic message corruption using OpenJDK that we didn't see with the Oracle JRE. Not to mention that the Oracle JRE is the authoritative implementation of Java which matters in a production environment. After that, we chose to use the Oracle JRE at least until we can circle back and retest and help fix OpenJDK.

We want 64-bit and we've chosen Oracle's JRE as the path forward. That means that we had to turn to Linux in order to run our Kafka, Cassandra, and Spark environments. As we continue to build out these services, this will result in us having more and more Linux.

System Tuning: My Server Isn't That Wimpy

Another area that could use more attention is out-of-the-box tuning for a FreeBSD install. While many defaults seem to be tuned for the absolute least common denominator of support, it'd be great to implement some way to have profiles that can be selected during install. Let me explain with concrete examples.

Right now, we compile our Tcl interpreter with an additional CFLAGS value of `-DFD_SETSIZE=4096` to override the default of 1024 from `<sys/select.h>`. We know that `select()` isn't ideal and we've put out a bounty for Tcl kqueue support, but at the same time 1024 seems really small for modern day net-

FlightAware and FreeBSD



work applications. It'd be great to see this increased so that FreeBSD systems could be ready to handle all of the concurrent connections that it can otherwise effortlessly support without further tweaking by the user.

Default log rotation sizes are also something that needs some attention. According to the `/etc/newsyslog.conf` on a fresh FreeBSD install, most log files should be rotated after they grow to be 100KB in size. I don't know about you, but my `/var/log/messages` and my `/var/log/auth.log` don't take very long to reach that. In an era where you can get 10TB hard disks and 1TB SSDs, this seems like an obvious candidate for a profile system with options for embedded, server, desktop, etc., so that the target rotation size is more reasonable. Even 100KB seems small for a SD card-based system.

While I'm complaining about logging, this seems like the opportune time to suggest some features. Can we get an include directive for `/etc/syslog.conf`? `newsyslog.conf` has one, so it seems reasonable that `syslog` should as well. Some improvements to the `!` program specifier in `syslog.conf` would be nice too. As it is, in order to log all messages from postgres to `/var/log/postgres.log`, I have to do something like:

```
# Capture PostgreSQL logs
!postgres
*. * /var/log/postgres.log

# Capture all other logs
!-postgres
*. * /var/log/all.log
```

It would be nice if there was a way to have a program specifier that matched anything except other defined program specifiers. Then you can have a catch-all section without having to list the exclusions out specifically. This will also work around another issue I encountered: the maximum line length of `syslog.conf` is too short for listing these out.

Finally, another area that could be improved is overall network stack tuning. If you search Google, you can find lots of different pages that tell you `sysctls` and `tunables` that you should be setting to maximize your network stack and 10GigE NIC throughput. It would be great if there was a profile selection during install that could tune things to be great for Varnish, nginx, Apache,

and so forth, as well as GigE, 10GigE, or 40GigE networking. As it is, it is sort of a dark art to get it set up and fully understand it.

Debugging: Because Things Break

What would you do if the `zpool` command dumped core? For me, I'd fire up the debugger and hope to see where it crashed by loading up the core file. Unfortunately, the base system lacks debugging symbols when you install a release version of FreeBSD. Why is this? Are they too big, just like those 100KB log files? It would be nice if base binaries weren't stripped, or at least an optional system component included symbol files for all of the base binaries and libraries.

Similarly, the ports tree defaults to building software without symbols. Most ports have a `DEBUG` option you can enable that will cause the software to be built with symbols but generally without any compiler optimizations enabled. This isn't ideal for production. I'd like to see a universal option to build a port with optimization but also with debugging symbols, ideally by default. I understand that the debugging will be a bit harder with something built using `-O` or `-O2`, but it is sure better than nothing.

Finally, everything should have `DTRACE` by default. `DTRACE` is an amazing tool that FlightAware has used on more than one occasion to understand a performance problem or system failure. Both PostgreSQL and Tcl conveniently have extensive `DTRACE` support. However, ports does not enable this by default. In our opinion, every port should default to building with `DTRACE`. It isn't needed on a day-to-day basis but, like debugging symbols, it is invaluable when a problem eventually does appear.

Wrapping Up

I hope I've been able to make it clear that FlightAware really does prefer and advocate for FreeBSD. That being said, there is always room for improvement and I set out to explain here what could be improved in FreeBSD to make it work better for us and presumably others. It is time to start looking at ways to make it easier to increase the number of installs in the wild and make it easier to treat FreeBSD as a commodity platform. ●

SEAN KELLY has been a member of the FlightAware team since 2012. He serves as the Director of IT Operations. In this role, Sean designs and oversees the growth and maintenance of both the worldwide infrastructure that powers FlightAware as well as all of the IT systems powering FlightAware's two corporate offices. Sean is also a former committer of the FreeBSD Project and has been using FreeBSD since at least the 4.0 days.

Support FreeBSD®



Donate to the Foundation!

You already know that FreeBSD is an internationally recognized leader in providing a high-performance, secure, and stable operating system. It's because of you. Your donations have a direct impact on the Project.

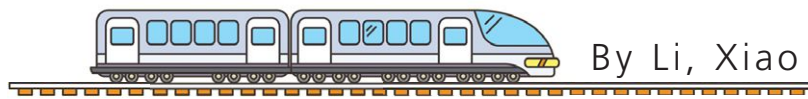
Please consider making a gift to support FreeBSD for the coming year. It's only with your help that we can continue and increase our support to make FreeBSD the high-performance, secure, and reliable OS you know and love!

Your investment will help:

- Funding Projects to Advance FreeBSD
- Increasing Our FreeBSD Advocacy and Marketing Efforts
- Providing Additional Conference Resources and Travel Grants
- Continued Development of the FreeBSD Journal
- Protecting FreeBSD IP and Providing Legal Support to the Project
- Purchasing Hardware to Build and Improve FreeBSD Project Infrastructure

Making a donation is quick and easy.
freebsd.foundation.org/donate





By Li, Xiao

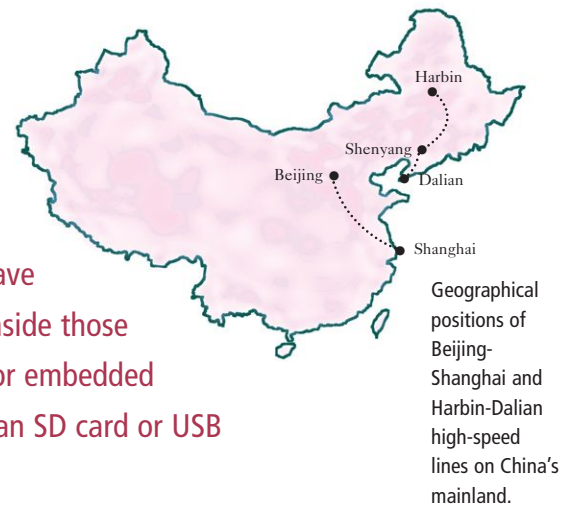
FreeBSD-powered LCD Advertising Displays

In 2011, 101 freestanding LCD advertising displays were installed in 11 waiting rooms along the high-speed railway between Shanghai and Xuzhou. In 2013, seven LCD advertising display towers were installed in four waiting rooms along the high-speed railway between Shenyang and Dalian. In these devices, tailored FreeBSD images with Xorg run with Intel CPUs and flash memories. The strong body of FreeBSD ensures that the advertising pictures and videos are stably played and that the electric motors in the towers behave correctly under control of the advertising programs.



Above: Freestanding LCD displays. Only the lower LCDs were driven by hosts running FreeBSD. They were playing advertising pictures customized by Mega-info Media.

Since 2000, electronic displays such as LCD and LED have been widely used in advertising all over the world. Inside those advertising devices, open-source software compiled for embedded systems supports playing audio and video, accessing an SD card or USB flash drive, and networking and remote control.



Since 2000, China has built the longest high-speed railway lines in the world [1]. In order to develop the economic value of waiting rooms in railway stations along those high-speed lines, LCD and LED displays have been installed to play commercial advertising pictures and videos.

From 2011 to 2013, I was an advertising equipment provider for some of those locations. And I adopted FreeBSD with Xorg as the operating system in my products. The stability of FreeBSD and my software product surprised many customers with their very rare software failures.

Products and Installations

I secured two equipment supply contracts in 2011 and 2013, respectively for a section of the Beijing-Shanghai high-speed line and a section of the Harbin-Dalian high-speed line. Map above shows the geographical positions of the two lines on China's mainland [17].

Beijing-Shanghai High-speed Railway Line

In the section between Shanghai and Xuzhou (about 626 kilometers [6]) of the Beijing-Shanghai high-speed railway line [2-3] (about 1318 kilometers [6]), the customer ordered the freestanding LCD advertising displays shown in the inset photo on page 10 [9]. They were installed in waiting rooms of the railway stations listed in Figure 1.

The hardware configuration to run FreeBSD is shown in Table 1 [7-8]. In 2011, the ethernet and the audio controllers had been fully supported by FreeBSD's drivers `re(4)` and `snd_hda(4)`. But the GPU was somewhat of a newbie for FreeBSD, and required a newer Xorg driver `xf86-video-intel29`, which didn't belong to the main line of Xorg dependencies

in the FreeBSD ports tree.

At that time, Xorg had imported a kernel mode setting (KMS) mechanism in the GNU/Linux community. But FreeBSD hadn't kept up with them. Fortunately, `x11-drivers/xf86-video-intel29` worked fine with Xorg 7.5.1 and the onboard GPU in user mode. The display mode could be smoothly set to `1920x1080p@60Hz`, which fully conformed to the monitors' best performance. And the X-Video extension of Xorg also worked fine with

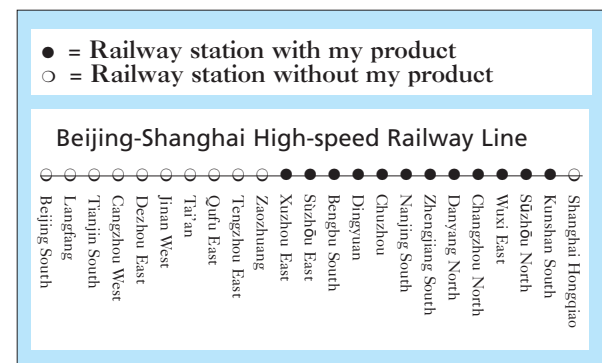


Fig. 1. The main line of the Beijing-Shanghai high-speed railway line. Since 2016, some of my products have been removed from some of those railway stations such as Nanjing South.

Motherboard	Intel Desktop Board D410PT or D425KT
CPU	Intel Atom D410 or D425
GPU	Intel GMA 3150
Chipset	Intel NM10
Ethernet	Realtek 8103EL or 8105E
Audio	Realtek ALC662
RAM	1GB or 2GB
Storage	8GB, USB flash drive

Table 1. Hardware configuration of my products for the Beijing-Shanghai high-speed railway line.

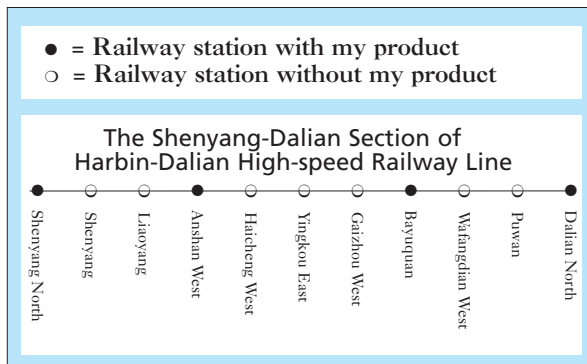


Fig. 2. The Shenyang-Dalian section of the Harbin-Dalian high-speed railway line.



LCD advertising display tower. It is playing advertising contents customized by Xinhua News Agency.

the driver, which prevented the CPU from being overloaded when playing video.

The Harbin-Dalian High-speed Railway Line

In the section between Shenyang and Dalian (about 380 kilometers [6]) of the Harbin-Dalian high-speed line (about 918 kilometers [6]), the customer ordered the LCD advertising display towers shown in photo at left [16]. They were installed in waiting rooms of the railway stations listed in Figure 2 [4-5].

The hardware configuration to run FreeBSD is shown in Table 2 [10-11]. In 2013, the ethernet controller Atheros GbE hadn't been supported by FreeBSD. But I didn't have enough time to hack the driver on my own. I had to install an additional PCI ethernet card with RTL8139 onto the motherboard.

In this application case, every host was designed to drive 5 LCD displays via VGA ports. Besides onboard GPU, I had to install two PCI-E graphics cards with VGA and DVI-I ports onto every motherboard. At that time, the GPU of NVIDIA worked fine with NVIDIA's closed source driver [12], but the Intel GPU could only be driven by xf86-video-vesa. Fortunately again, since the Xinerama extension of Xorg was applied to combine five LCD displays as a single virtual display, the XVideo extension wasn't necessary (actually, Xinerama disables most hardware rendering); xf86-video-vesa was sufficient for this case.

Technical Points

Tailored FreeBSD images with Xorg and my software were written into a USB flash drive or SATA SSD.

Tailored FreeBSD File Set

To simplify the work, an MBR partition table was created in every USB flash drive or SATA SSD. Only one partition was created in every drive, and a BSD label "a" was created in the partition. The slice /dev/da0s1a (for USB flash drive) or /dev/ada0s1a (for SATA SSD) was formatted into FreeBSD FFS 2 with soft updates and journaling.

In actual use, writing operations to drives was not frequent, at no more than about 100 MB a day. But abnormal power-off might occur 10 times a day. Thus, the robust filesystem could effectively reduce booting failure. Having benefited from FreeBSD FFS 2 with soft updates and journaling, booting failures of my products have

Motherboard	GIGABYTE GA-B75-D3V
CPU	Intel Core i5 (LGA1155)
GPU	Intel HD Graphics 2500 (onboard) NVIDIA GeForce 210 (PCI-E, doubled)
Chipset	Intel B75
Ethernet	Atheros GbE (onboard, unused) Realtek RTL8139 (PCI)
Audio	Realtek ALC887
RAM	2GB
Storage	SATA SSD 120GB, Intel

Table 2. Hardware configuration of my products for the Harbin-Dalian high-speed railway line.

been caused by hardware damage only recently.

The slice `/dev/da0s1a` or `/dev/ada0s1a` is used as the root filesystem of FreeBSD. The hierarchy is shown in Table 3. Since I found “mfsBSD” made by Martin Matuška [13], I have looked through his work for references.

Tailored Xorg File Set

To run the GUI, a file set of Xorg had to be written into the filesystem. The hierarchy is shown in Table 4.

I hadn’t integrated any typical X11 toolkit such as GTK into the flash drive.

Use of wxWidgets

wxWidgets [14] has been collected in the FreeBSD ports tree, which is compiled with GTK 2, but wxWidgets can work directly with X11. The port wxWidgets with X11 is called wxX11. Although wxX11 is incomplete and buggy so far, it can well support picture file handling and basic sub-window management which is required for playing advertising contents.

For my product, wxWidgets 2.8.x can be configured as:

```
./configure --with-x11 --with-opengl\
--disable-shared --enable-unicode
```

After compiling wxX11 (GNU make is required), the script `wx-config` can be used to get C++ compiling and linking parameters for the header and library files of wxX11. The commands look like:

```
# To compile C++ source files
~/wxWidgets-2.8.12/wx-config --cxxflags
# To link object files
~/wxWidgets-2.8.12/wx-config --libs
~/wxWidgets-2.8.12/wx-config --gl_libs
```

Motor Control via RS-232-C Port

To simplify my program, `stty(1)` was used against `/dev/ttyu0.init` to set default baud rate, bit count, and other options before my program opened `/dev/ttyu0`.

The onboard serial port was connected with a printed circuit board (PCB) I designed to set the electric motors’ behaviors: start and stop, rotating direction, and rotating speed. At the same time, some physical quantities (e.g., ambient temperature and angular displacement) captured by sensors on my PCB were sent to FreeBSD via the serial port.

<code>/boot/</code>	FreeBSD kernel and driver modules, loader(8) and configuration files
<code>/libexec/</code>	Run time link-editor, <code>ld-elf.so.1</code>
<code>/bin/</code> , <code>/sbin/</code> , <code>/usr/bin/</code> , <code>/usr/sbin/</code>	Command line utilities (e.g. <code>/bin/sh</code> , <code>/bin/rm</code> , <code>/usr/bin/killall</code> , <code>/sbin/ifconfig</code> and <code>/sbin/mount</code>)
<code>/dev/</code>	An empty directory for <code>devfs(5)</code>
<code>/var/</code> , <code>/tmp/</code>	Multipurpose files (e.g. <code>/var/empty</code>)
<code>/etc/</code>	Initializing scripts and configuration files
<code>/lib/</code> , <code>/usr/lib/</code>	Shared objects
<code>/usr/local/</code>	Xorg
<code>/root/</code>	My own software files

Table 3. The outlined layout of the filesystem in the USB flash drive or SATA SSD.

Networking

SSH daemon configured with public key authentication was started in system. SSH can supply secure and multipurpose channels for various remote accessing. An `rsh(1)`-like remote command executing via `ssh(1)` simplifies networking into writing some small `/bin/sh` scripts, avoiding the designing of a tiring and buggy transport protocol.

SSH can also act as a secure transport mechanism of `rsync(1)`, which is used to transfer large files.

A Customized Script `/etc/rc`

After the FreeBSD kernel is started, usually `/sbin/init` is executed, which tries to run `/etc/rc` to do most of the essential initialization [15]. My customized script `/etc/rc` contained initializing steps

<code>bin/</code>	Standalone executable files (e.g. Xorg, <code>xrandr</code> and <code>xkbcomp</code>)
<code>etc/</code>	<code>xorg.conf</code> and configuration files for font-config and pango
<code>lib/</code>	Shared objects
<code>lib/X11/fonts/</code>	Basic fonts
<code>lib/dri/</code>	DRI drivers
<code>lib/pango/</code>	Pango modules
<code>lib/xorg/modules/</code>	Drivers
<code>share/X11/xkb/</code>	Resource files concerning keyboard

Table 4. The outlined file layout of tailored Xorg under `/usr/local/`. I hadn’t integrated any typical X11 toolkit such as GTK into the flash drive.

reduced from a complete FreeBSD base and a complete Xorg distribution. At the end of the script, it started my own program. The main line of the script is:

1. Waiting for the USB device to get ready in several seconds.
2. Finding and checking the storage device in a possible set of `/dev/da0s1a` and `/dev/ada0s1a` and mounting it in read-write mode.
3. Finding and configuring the network interface in a possible set of `re(4)`, `rl(4)`, and `em(4)`. Also setting network routing. The driver `em(4)` is used when debugging the file set in VirtualBox.
4. Adjusting the time zone setting of the kernel by `adjkerntz(8)`.
5. Tuning the sound volume.
6. Starting the SSH daemon.
7. Starting the Xorg server.
8. Starting my own program.

Proposals

In the source tree of FreeBSD, there is a great thing—`ndis(4)`. It can run the network device driver written for Microsoft Windows inside the FreeBSD kernel.

But the module lacks maintenance. It often doesn't work with up-to-date device drivers for Windows, especially wireless ethernet device drivers.

What's more, `ndis(4)` can only cover quite a

few of the network devices. Actually, in reality, although FreeBSD has a very robust and well-designed kernel, it leaves people close to GNU/Linux because of its shortcomings.

For the future of FreeBSD, my proposal has to do with the device driver:

1. Move the Windows kernel compatibility layer of `ndis(4)` from the FreeBSD kernel to user mode, which means the layer and device driver will run in a process. In user mode, all programs are easier to hack and debug than in kernel. At least most faults in user mode don't cause kernel crash.
2. Extend the coverage of the compatibility layer, which means covering more network device drivers and trying to cover other types of device drivers such as that of the GPU.

Acknowledgment

Thanks to Liang, Li (also known as Kylie Liang) for her help with this paper. ●

Li, Xiao is a software and hardware engineer living in China. He runs his tiny company in Beijing. He is an experienced FreeBSD developer. He worked on ports of **LaTeX-CJK** and **Linux compatibility** in 2006. He is enthusiastically involved in the FreeBSD community of China. He is interested in design of printed circuit board and cross-platform software.

REFERENCES

- [1] "High-speed rail in China," Wikipedia, online: https://en.wikipedia.org/wiki/High-speed_rail_in_China.
- [2] "Beijing–Shanghai High-Speed Railway," Wikipedia, online: https://en.wikipedia.org/wiki/Beijing%E2%80%93Shanghai_High-Speed_Railway.
- [3] (In Chinese Only) "Beijing–Shanghai High-speed Railway," Baidu Encyclopedia, online: <http://baike.baidu.com/view/141756.htm>.
- [4] "Harbin-Dalian High-Speed Railway," Wikipedia, online: https://en.wikipedia.org/wiki/Harbin%E2%80%93Dalian_High-Speed_Railway.
- [5] (In Chinese Only) "Harbin-Dalian High-speed Railway," Baidu Encyclopedia, online: <http://baike.baidu.com/view/1213823.htm>.
- [6] (In Chinese Only) High-speed Railway Web, online: <http://www.gaotie.cn/>.
- [7] "Technical Product Specification for the Intel® Desktop Board D410PT," Intel, online: <http://www.intel.com/content/www/us/en/support/boards-and-kits/desktop-boards/000021877.html>.
- [8] "Technical Product Specification for Intel Desktop Board D425KT/D425KTW," Intel, online: <http://www.intel.com/content/www/us/en/support/boards-and-kits/desktop-boards/000020992.html>.
- [9] Mega-info Media Co. Ltd., the official website: <http://www.megainfomedia.com/>.
- [10] GA-B75-D3V User's Manual, GIGA-BYTE Technology Co. Ltd., online: http://download.gigabyte.cn/FileList/Manual/mb-manual_ga-b75-d3v_v1.2_e.pdf.
- [11] Introduction to Intel Core i5-3450, Intel, online: http://ark.intel.com/products/65511/Intel-Corei5-3450-Processor-6M-Cache-up-to-3_50-GHz.
- [12] GeForce driver for FreeBSD, NVIDIA, online: <http://www.geforce.com/drivers>.
- [13] Martin Matuška, "mfsBSD", online: <http://mfsbsd.vx.sk/>.
- [14] wxWidgets, a cross-platform C++ GUI library, online: <http://www.wxwidgets.org/>.
- [15] "Kernel Initialization," "FreeBSD Architecture Handbook," online: https://www.freebsd.org/doc/en_US.ISO8859-1/books/arch-handbook/boot-kernel.html.
- [16] Xinhua News Agency, China, the official website: <http://www.news.cn/english/>.
- [17] A reworked profile map of China's mainland, referenced from the governmental digital map website "Tianditu" (The Chinese meaning of this name is "Sky-land Map" or "Sky-earth Map"), online: Chinese: <http://map.tianditu.com/> English: <http://en.tianditu.com/>.

THE INTERNET NEEDS YOU

GET CERTIFIED AND GET IN THERE!
Go to the next level with



Getting the most out of
BSD operating systems requires a
serious level of knowledge
and expertise



NEED AN EDGE?

- **BSD Certification can make all the difference.**
Today's Internet is complex. Companies need individuals with proven skills to work on some of the most advanced systems on the Net. With BSD Certification

**YOU'LL HAVE
WHAT IT TAKES!**

SHOW YOUR STUFF!

Your commitment and dedication to achieving the **BSD ASSOCIATE CERTIFICATION** can bring you to the attention of companies that need your skills.



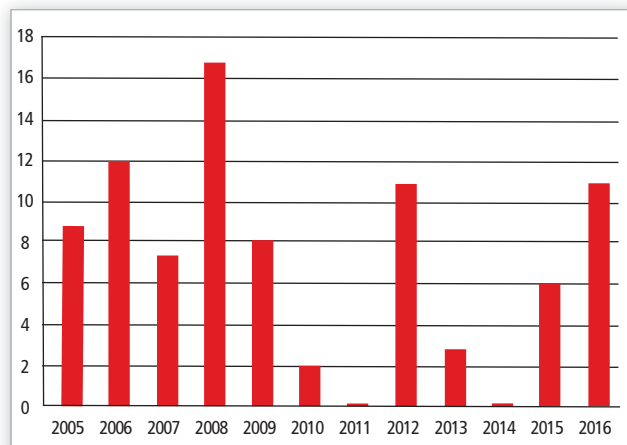
BSDCERTIFICATION.ORG

Providing psychometrically valid, globally affordable exams in BSD Systems Administration

Improving the FreeBSD Translation Tools

By Warren Block

The first question that comes up about translation is “Why bother?” Or maybe, to put it more politely, is the effort of translating worth the result? A large portion of the world is inhabited by people who could benefit from using FreeBSD—and can’t—because they don’t speak English even as a second or third language. There is a huge benefit that we can bring to them.



Year	New translations
2005	9
2006	12
2007	7
2008	17
2009	8
2010	2

Year	New translations
2011	0
2012	11
2013	3
2014	0
2015	6
2016	11

Secondly, translators are usually the ambassadors into a new area, region, or country. They are not just translators; they use whatever they have translated. If you can make the job easier for translators, you can also increase the number of people who are using your system in those areas.

Finally, you can enlarge the community. Metcalfe’s Law, roughly paraphrased, states that the value of a telecommunications network is proportional to the number of nodes in that network. In other words, the more nodes, the more valuable it is. With the Internet, of course, if you had only two nodes on it, it would not be that useful. But if you have a million, it’s much more useful. And I say that FreeBSD is a telecommunications network, and people are the nodes in that network. More users mean more people discovering bugs, submitting patches, adding features, enhancing documentation, and all this increases the value of FreeBSD for the very same people who are doing that work. It’s a feedback loop. That is the value of translation.

The FreeBSD Documentation

There are several different categories of FreeBSD documentation. There are the books and articles, which are

marked up in DocBook XML. That is not just the famous *FreeBSD Handbook*, but also other books like the *Porter's Handbook*, which describes how to port programs to FreeBSD. It's a large, detailed, continuously updated book that is well worth a read. There are also a number of stand-alone articles marked up with DocBook XML. Using DocBook XML gives us the ability to render that source into a number of output formats like HTML, PDF, ePUB, PostScript, or even plain ASCII.

Secondly, we have our man pages. Ten or twenty years ago, some of the open-source world didn't quite see the advantage of man pages, or didn't understand it, and kind of took a wrong turn. But for the BSD community, this is a major feature. A man page is not a tutorial; it's meant to be a quick reference for when the exact format of a command or some configuration detail can't be remembered. You can look in the man page and there it is. We have a huge resource in those man pages, and they are continuously updated and improved. They use the `mdoc(7)` markup language, mostly, and there are some old ones that use `roff` or `troff`.

There is other documentation. Part of the FreeBSD Documentation Project's domain includes the source. If there is an error in spelling or grammar or clarity in source files, we are allowed to edit those. This also holds for text files, or any file, really. That is the scope of the FreeBSD Documentation Project.

Whitespace

One surprisingly difficult problem we encounter with translation is whitespace. These are the blank areas between text: spaces, tabs, line feeds, carriage returns, and a number of others. Being invisible, these characters are hard to see, and this makes them difficult to explain. The reason they matter in translation is that when a document is edited, translators have no easy way to tell if the content changed and a new translation is required, or if only whitespace changed and the existing translation is still valid. A typical example is when someone rewraps a paragraph to fit a particular line-length in an editor. The content is still the same, but that is difficult for translators to see without rereading the whole thing. At worst, the translator might retranslate the document unnecessarily. That is very demotivating and a serious problem when we depend on volunteer translators.

Whitespace changes have traditionally been managed in the same way as is usually done with source code. Changes to content are done in one

commit; then a second commit changes whitespace—and only whitespace—to modify formatting. Whitespace-only changes to documents usually have a commit message that points out that content has not changed and the document does not need any new translation.

Separating content and whitespace commits helps translators, but adds a serious amount of work for people editing the original documents. Only content or whitespace can be changed at a single time, so documents become ragged and poorly formatted. When working on the follow-up, whitespace-only change, errors in content are often found. These cannot be fixed immediately, but require another round of content change and then yet another whitespace-only change. So, whitespace is a problem for pretty much everyone working on documentation.

The Old Translation Method

The traditional FreeBSD document translation method is simple in concept: translators work by commits. Translators work through the changes made to the English documents, translating those changes into the target language.

In practice, this method is difficult and time-consuming for translators. It is fully manual, giving no assistance to the translator. Quite the opposite, in fact: translators must work through commits in order, essentially having no control over the size of a change to the original document. A single sentence might have changed, or a whole chapter rewritten. Until that change has been translated, subsequent changes cannot be translated, and this can prevent other translators from working on the document. The scale of a change is also a problem with volunteer translators, who might see a large change as too much work for the time they are willing to contribute.

There is no formal method, but translation teams typically keep track of the latest translated version of a document with a comment noting the last commit number translated from the English document. This, too, is completely manual.

To locate and work by commits requires a non-trivial familiarity with the version control system. Because translators are working directly with DocBook XML source files, a non-trivial familiarity with DocBook is also often required. The quality and quantity of translations created with this old method is a testament to the dedication of the translation teams.

After all this overhead, the translator has finally located the next change to the English document



Translation Tools

that must be added to the translated version. This change could be anything: new text added, old text removed, or both. At some point, the translator will compare the previous translated document with the diff file containing the changes. Because this method is fully manual, there is no assistance in locating the place in the document source where the changes occurred. Translations do not match the English documents line-for-line, so the translator is forced to locate the place where changes must be made. This can end up taking a lot of time when the change to the English document was non-trivial.

Finally, the translator can get to the one thing they wanted to do: creating a translation of the English document in the target language. Prospective translators do not respond well to this demotivating, labor-intensive workflow.

The New Translation Method

Something had to change. In 2012, Thomas Abthorpe and Benedict Reuschling were talking about a new translation method using "PO files." Even though I am a monolingual American, this work was obviously very important, and I slowly learned about ways to use this new translation method. In 2015, we finally had a working system.

The new translation method uses gettext, a program developed so vendors would not have to recompile programs so that their messages could be in a different language. Instead, a Portable Object (PO) file was created that contained both the English strings and their translated equivalents. After the user defined their locale, the strings were shown in the local language.

At first glance, this does not seem like a system that would lend itself to translating entire documents, but it does turn out to work surprisingly well for that. A program is used to extract strings from the original English DocBook XML source into a PO file. Translators use a PO editor to edit these files. The English string is shown, and the translator enters the equivalent translated string next to it. There is a 1:1 correspondence between the original and the translation. The translator does not have to look through source files to determine where the changes must be made. Whitespace goes from being a serious problem to mostly not being an issue. Finally, there is a "translation memory" that offers to reuse translations known from earlier uses. This is

neither as good nor as bad as people tend to imagine, but it can help reduce the translator's workload by perhaps 5–15%.

The new translation method has only three steps:

1. Run 'make po' to extract translatable strings from the English document. If a translation already exists, it is preserved and updated with strings that have changed.
2. Run a PO editor and enter translations next to the English strings.
3. Run 'make tran' to build the translated version of the original document.

That is the entire process, all of it. The translator does not need extensive knowledge of the version control system or DocBook markup. They do not have to translate a particular change or entire large changes. Instead, they can do as much or as little translation work as they want without impeding other translators. Automation assists with the translation where possible, and PO editors typically show how much of the document has been translated. It is a huge change from the old translation method and lets volunteer translators actually contribute rather than driving them away.

Implementation

There are several programs that can extract translatable strings from XML files. The one chosen for this project was itstool (<http://itstool.org>), because of its simplicity and standards compliance. Some small utilities from the gettext-tools package are also used. itstool requires Python, which is already a required dependency of the documentation port (textproc/docproj). The gettext-tools port (devel/gettext-tools) is also likely already installed on a system used by a documentation writer or translator, so the overhead of these tools is very small.

Outcome

The new PO translation system went live at the end of August 2015. In the remainder of 2015, there were six newly-created translations of articles and books into German, Dutch, Spanish, Chinese, and Korean. This compares favorably to the numbers for 2013, when there were only three new translations, and 2014, when there were no new translations at all. In 2016, there

have been another 10 new translations, including two very large translations of the *Handbook* and *Porter's Handbook* into traditional Chinese. These numbers admittedly disregard the size of new translations and ongoing maintenance on existing translations. However, they demonstrate that simplified translation methods can be an enabling technology, and that given better tools, volunteers are capable of producing large quantities of useful work.

Challenges

We do still have some challenges, and I call these "challenges" because when you have a problem, it's just a problem. When you have a challenge, you are challenged to come up with a solution.

We have things that should not be translated. A prime example is our article that contains developer PGP encryption keys. That is all it contains, two sentences of introduction (essentially, "Our developers have PGP keys. Here they are.") and then 600 pages of PGP keys. These are just numbers. We don't want these translated, because the translated form of them is identical to the original. It wastes the translator's time by

even showing them these strings. But we also want a single source for this type of information. If these strings were translated, they would be copied into the translated file, making multiple copies of them and guaranteeing that some will always be out of date in the translated version. So, we need a way to mark up the source to say "this string should not be translated." Ideally, the translator would never even see that string. Instead of seeing 600 pages of XML source, they would only see the two sentences of introduction.

Searches for standard ways of marking strings that should not be translated did not result in any obvious way of accomplishing that. There were a few organizations that had used toolchain-specific methods that appeared to not be applicable to our documentation. A messy hack of using XML processing instructions might be workable with our toolchain, but a better way is still preferable. The search is still on, and suggestions are always welcome.

Another challenge is preserving the huge amount of work put into translations that were created with the old translation method. Some of

RootBSD

Premier VPS Hosting

RootBSD has multiple datacenter locations,
and offers friendly, knowledgeable support staff.
Starting at just \$20/mo you are granted access to the latest
FreeBSD, full Root Access, and Private Cloud options.



www.rootbsd.net

these are very current, and we want to preserve as much of that work as possible. Ideally, we would convert those translations into PO translations without losing any of that work. The GNOME xml2po program can take an English original document and a fully translated version and produce a PO file from them. However, the two documents must correspond line to line exactly. But our translation teams have their own rules for line wrapping, so the original and translation do not match. There might be other programs out there to do this based on matching the XML elements of the two files. It is likely that not much effort has been put into conversion programs because conversion for most organizations is a one-time occurrence.

Documentation translators would benefit from seeing a few lines of text from before and after the source string to get an idea of the context in which it is used. Many PO editors do not show much context. The original use of gettext was for translating program prompts where there was little or no context to show. With documentation, that is very different, and the job of translating is easier if surrounding context is shown. This is not a technical problem, because the PO files already have a comment with each line showing its line number from the original XML file.

There are many PO editors, but only a few have been ported to FreeBSD. Currently, there are three: editors/poedit, devel/gtranslator, and devel/lokalize. Having more PO editors in ports will make it easier for translators to choose an editor that is well-suited to translating the specific document or target language. A nearly functional port of the well-regarded Virtaal (<http://virtaal.translatehouse.org/>) is available and probably does not need much more effort to complete. Java-based PO editors are available, and the web-based online Pootle system is also in ports as textproc/pootle. PC-BSD had used Pootle for their translations. PC-BSD has now been renamed TrueOS, and currently uses Weblate (<https://weblate.org/en/>) for web-based translation. There are also commercially hosted sites with similar operation, like Transifex (<https://www.transifex.com/>), which is free for open-source use.

Potential

The most obvious and compelling possibility with PO-based translations is translation of the FreeBSD man pages. While itstool is strictly for XML files, the Debian po4a package is capable of extracting strings from man pages to PO files. A

fully translated set of FreeBSD man pages would be incredibly valuable. With the barriers to translation lowered by the PO system, this becomes possible.

New translators often become FreeBSD documentation contributors and committers. PO translation offers an easy way to begin contributing and an introduction to the community which can lead to increased participation.

New translations of documentation will bring in new users from new regions and countries, offering their own unique perspective on problems and opportunities for the community. Some of those people will continue on to become contributors, making FreeBSD better for everyone who uses it.

What We Learned

The journey to make PO translation tools usable on FreeBSD taught us some related things.

The value of cross-pollinating with other projects should not be underestimated. Some of the most valuable insights were gained when Ryan Lortie of the GNOME project and I were talking before or after presentations at BSDCan. Different projects have different experiences, and two or more projects can benefit by sharing their knowledge. We should not look at other projects as competitors, but as rich gold mines of information.

The old translation method is essentially a commit-by-commit porting effort, porting an English document to another language. When the X11 team asked about commit-by-commit versus file-by-file porting, we were able to identify some of the problems inherent to the commit-by-commit method:

- commits must be done in order, even if a critically important later commit is needed
- usually the porter is forced to work in units of an entire source commit, regardless of how large it might have been, and the project can be stalled while one porter struggles with a large commit
- even though earlier commits can be entirely erased by later ones, the work to port them must still be done because later commits depend on their presence.

Technical debt can be a serious problem. When we tell people how we include chapters in XML files by defining them as entities, and they

pause and say, "We didn't think anyone was still doing it that way," it is a warning sign. Even if we follow standards, there is a danger that the new tool might not support the old methods. Ultimately, this could leave documentation or translation work stranded while new methods are implemented. Keeping up-to-date with industry standards is usually less painful if it is done gradually rather than as a sudden, urgent need.

Related to the previous point about technical debt: our documents need to be switched to UTF-8. It is long past time, and I have written a program to do this. All that remains is the hard work of verifying that the conversion works correctly.

Acknowledgments

The work of bringing the PO translation tools to the FreeBSD documentation has been under way for several years. Many people have contributed, both from within and outside the FreeBSD Project. This list is not complete, although anyone missing is due to forgetfulness on my part. My sincere thanks to everyone for their contribution to making FreeBSD more available to people around the world!

Thomas Abthorpe
Glen Barber
Mark Felder
Hiroki Sato
René Ladan
Ryan Lortie


Koop Mast
Shaun McCance
Chris Petrik
Benedict Reuschling
...and many others.

Links

Material discussed in this article, including the presentations, scripts used to collect statistics, graphs, and the Virtaal port, is here:


<http://wonkity.com/~wblock/translation/>



WARREN BLOCK has been using FreeBSD since 1998 for such diverse things as servers, documentation creation, network monitoring, and enterprise computing. He has been a FreeBSD documentation committer since 2011 and a member of the Documentation Engineering team since 2014. In 2016, he began working on the FreeNAS documentation for iXsystems Inc.



Rack-mount networking server

Designed for BSD and Linux Systems
Up to **5.5Gbit/s** routing power!

Made for  FreeBSD


PERFECT FOR


- ▶ BGP & OSPF routing
- ▶ Firewall & UTM Security Appliances
- ▶ Intrusion Detection & WAF
- ▶ CDN & Web Cache / Proxy
- ▶ E-mail Server & SMTP Filtering
- ▶ Anti-DDoS and clean pipe filtering


KEY FEATURES


- ▶ 6 NICs w/ Intel igb(4) driver w/ bypass
- ▶ Hand-picked server chipsets
- ▶ Netmap Ready (FreeBSD & pfSense)
- ▶ Up to 14 Gigabit expansion ports
- ▶ Up to 4x10GbE SFP+ expansion

I Gbit/s Copper	Ports	Chipset
L800-G808-1	8x Gbe RJ-45 ports	8x Intel i210 AT; PEX8618
L800-G808-2	8x Gbe RJ-45 ports	8x Intel i210 AT; PEX8618
L800-G428-1	4x Gbe RJ-45 ports	1x Intel i350 AM4
L800-G428-2	4x Gbe RJ-45 ports	1x Intel i350 AM4
I Gbit/s SFP (Fiber)	Ports	Chipset
L800-S406-1	4x Gbe SFP ports	i350-AM4
10GbE Copper	Ports	Chipset
L800-T202-1	2x 10Gb RJ-45 ports	Intel X540
L800-T203-1	2x 10Gb RJ-45 ports	Intel X540
10GbE SFP+ (Fiber)	Ports	Chipset
L800-X204-1	2x 10Gb SFP+	Intel 82599ES
L800-X205-1	2x 10Gb SFP+	Intel 82599ES
L800-X405-1	4x 10Gb SFP+	Intel 82599ES; PEX8724









Designed. Certified. Supported

contactus@serveru.us | www.serveru.us | 8001 NW 64th St. Miami, FL 33166 | +1 (305) 421-9956



The Foundation's **Global Efforts**

The FreeBSD Foundation is proud to serve a welcoming international community. Part of our mission is to support and grow FreeBSD awareness throughout the world. To that end, our Board of Directors is made up of members based around the globe, and they are dedicated to helping us spread the word. One of the ways we do this is by sponsoring and attending conferences. In addition to sponsoring BSD-related events such as EuroBSDcon, AsiaBSDcon, BSDCan, and the Cambridge FreeBSD Developers Summit (BSDCam), the Foundation also participates in non-BSD events throughout the world, including womENcourage in Austria, Chemnitz Linux Days in Germany, FOSDEM in Brussels, and many more.

Last year, with the help of new board member Kylie Liang, we were able to begin extending our reach to new regions. Kylie arranged for members of the Foundation Board to present at a number of Chinese conferences and meetups to increase the awareness and adoption of FreeBSD in China. First, at OSC 2016: The Annual Open Source Conference, Foundation Board Director George Neville-Neil presented "FreeBSD is Not a

Linux Distro" to around 400+ attendees. Next, Foundation Board Director Hiroki Sato presented an "Introduction to FreeBSD" to around 300 attendees at COSCon 2016: The China Open Source Conference. This was the first annual conference sponsored by the open-source organization Kaiyuanshe. Following the presentation, 30 people joined Hiroki in a WeChat group for BSD. In addition, Director Robert Watson delivered a presentation on "Cambridge L41: Teaching Advanced Operating Systems with FreeBSD" through audio conference, and FreeBSD src committer Yanmin Qiao presented "FreeBSD and BSD-based Virtual Appliance in Microsoft Azure" at a BSD meetup in Shanghai. Finally, approximately 45 people attended a BSD meetup in Beijing. George Neville-Neil again delivered his "FreeBSD is Not a Linux Distro," Zheng Fu of Array Networks shared "FreeBSD development @ Array," and Yanmin Qiao talked about "FreeBSD and BSD-based Virtual Appliance in Microsoft Azure." More on our recent efforts in China can be found in our December 2016 newsletter: <https://www.freebsd.foundation.org/wp-content/uploads/2016/12/FreeBSD-Foundation-December>

Not only does the Foundation's own team attend events, but our **TRAVEL GRANT PROGRAM** allows for other community members to do so as well.



Photo by Ollivier Robert
Photo by Ollivier Robert

[2016-Update.pdf](#)

This year, we're working with new board member Philip Paeps to bring a greater FreeBSD presence to India by sponsoring Rootconf 2017, taking place in Bangalore in May. In addition to having a developer's room at FOSDEM in February, Director Benedict Reuschling will be heading up a booth as part of the event's exhibition, allowing us to better introduce the FOSDEM attendees to FreeBSD. George and Benedict will be teaching a two-week FreeBSD OS class in Darmstadt, Germany, and Benedict will also be teaching a course in Oulu, Finland.

Not only does the Foundation's own team attend events, but our Travel Grant Program allows for other community members to do so as well. The program is designed to both bring members of the FreeBSD community face-to-face for further development of the Project, and to spread the word about FreeBSD. Travel grants are available to community members who need assistance with expenses to attend conferences related to FreeBSD development and advocacy. Due to the limited size of the Foundation board and staff, we rely on the FreeBSD community to attend the events that we cannot. You can find out more about our travel grant program here: <https://www.freebsd.foundation.org/what-we-do/grants/travel-grants/>.

Attending events or speaking at meetups are just some of the ways the FreeBSD Foundation helps to increase global awareness of the FreeBSD Project. But, we can't do this without you. If there's an event you'd like to present at but are not sure how to go about it, please let us know. We can help. If you're looking for a speaker for your local meetup, we can help with that too. We've provided PDFs of our advocacy materials here: <https://www.freebsd.foundation.org/what-we-do/education-advocacy/>.

We can also provide raw files should you be interested in getting them translated into your preferred language. Finally, if you are a FreeBSD user in an area you feel lacking FreeBSD advocacy, contact us (marketing@freebsd.foundation.org).

Thank you for your continued support of the FreeBSD Foundation. ●

Anne Dickison is the Marketing Director for the FreeBSD Foundation. She spent over 15 years in technology-focused marketing and communications enjoying roles in biotech, publishing, and the nonprofit industry. When not promoting FreeBSD and the importance of open-source technologies, you'll find her enjoying game nights with friends and family and chasing after her wee dog Finnegan.



WELCOME to AsiaBSDCon 2017!

DATE

March 9–12, 2017

LOCATION

Tokyo University of Science
Tokyo, Japan

<https://2017.asiabsdcon.org>





new faces

of FreeBSD

BY DRU LAVIGNE

This column aims to shine a spotlight on contributors who recently received their commit bit and to introduce them to the FreeBSD community. This month, the spotlight is on **Nikolai Lifanov** (ports) and **Konrad Witaszczyk** (src), who became committers in November 2016, and **Larry Rosenman** and **Jean-Sébastien Pédrón**, who both received a ports commit bit in January 2017.

Tell us a bit about yourself, your background, and your interests.



Nikolai: I am originally from Moscow but have made a life for myself in North Carolina. I'm a systems/ops guy by trade. I enjoy collecting computers and playing mostly open-source rogue-like games. I also enjoy chess and origami.



Konrad: My name's Konrad Witaszczyk. I'm 25 years old and I'm a software developer at Wheel Systems where I work on Fudo PAM, a product based on FreeBSD. I studied Theoretical Computer Science at the Jagiellonian University. Currently, I live in Copenhagen, but in a few months I'll be back in Warsaw. My main interests are algorithms, cryptography, and security.



Larry: I've been a computer "nerd" since I was in 10th grade in high school, which was 1971. I've worked on all classes of computers (mainframes, minicomputers, and now microcomputers) up and down the entire software stack. I'm 59 years old, married, with one married 24-year-old daughter. As of February 13, 2017, I'll be working for MobileIron, a mobile device management company. Prior to this move, I worked for Alcatel-Lucent, which was bought by Nokia in 2016. I've also worked for IBM, Lombardi Software, Surgient, Pervasive Software, and others in various places around the U.S. I'm a big fan of open-source software, including FreeBSD,

PostgreSQL, and others. My hobby and my employment coincide, as computers are my hobby. I also do volunteer work for the American Red Cross on their Disaster Services Technology and IT End-User-Support teams.

Jean-Sébastien: Hi! I'm Jean-Sébastien Pédrón, 36 years old. I come from a small village in Brittany, in the west of France. I now live in Paris where I moved to work in IT in 2000. While in Brittany, I became interested in computers about the age of 14 or 15. Before that, I was spending (and still do) my spare time playing music (organ and percussions), fishing for carp, and observing planets and the deep sky in my local astronomy society. Now that I'm in Paris, astronomy or fishing has proved to be more difficult to do, so I started to practice rollerskating to compensate.



My nickname, "Dumbbell," comes from the following: Dumbbell, or M27 (27th object in the Messier catalog), is a nebula which is among the first deep sky objects you learn to locate and observe in the Northern Hemisphere. The first time I tried to find it on my own with a telescope, I needed an hour and a half, the crappiest performance of all time. The second time, it took me seconds, which is way more common. Ever since that time, the other members gave me that name and I kept it outside of the astronomy society as well.

When my family bought a computer, an IBM powered by an Intel 8086 with a 20 MiB

hard disk and a 3.5-inch floppy disk, I started to play with DOS and Basic. But I didn't do a lot. It's only when I began university in 1998 that I started to learn things. It's not at the university that I actually learned computer science, but with two friends I met there who were already deep into computer science and the Free Software world. That's the time I was introduced to C programming and Slackware Linux.

.....

How did you first learn about FreeBSD and what about FreeBSD interested you?

Nikolai: I learned about FreeBSD from a poll on a Linux forum about which Linux distribution people like best. FreeBSD was one of the options, and someone helpfully mentioned that FreeBSD is not Linux. I tried it out and it took me long enough to get my wireless card working to appreciate the unity of base system and documentation, the purity and simplicity of the BSD license, which focuses on protecting the code rather than the product, and that the community as a whole is more capable and excited than the Linux community. Perhaps unique to FreeBSD, developers have a wide range of skills and collectively care about the quality of the product as a whole. Have you written a cool utility, but mandoc markup needs some love? There is probably someone who can help with that. This is a sharp contrast to Linux systems where, for example, iproute2 maintainers may not care what the device mapper stack looks like.

Konrad: I first heard about FreeBSD from my brother. He experimented with various UNIX-like operating systems during his studies. Once he bought some magazines that included FreeBSD 5.4 CDs and decided to use it as his desktop. By the time I got interested in computers, he was running FreeBSD 7.0 on his laptop. I was 16 back then and really liked that he was managing his OS using a console and that he searched and installed programs using ports. The directory structure was clear and the documentation was easy for me to read. I liked that, more or less, a user knew what was happening in its OS instead of using a black box. Since then I've been using FreeBSD as my desktop and server.

Larry: I learned about FreeBSD back during the Internet .com boom/bust in the late 1990s. I was intrigued by the engineering of kernel, userland,

and the ports system all being engineered as a cohesive system, unlike Linux where it's which distribution, which version of which library, etc.

Jean-Sébastien: I don't remember exactly when I first heard about FreeBSD. I think it was through either *Linux Mag France* (a paper magazine) or <http://linuxfr.org>. I tried FreeBSD 4.6 on my home server. It maintained my Internet connectivity and hosted various services such as a website, a mail service, or a DNS. The system was so easy to use and comfortable with the unique `/etc/rc.conf` configuration file, and had such great documentation. The Ports system was great, and I wasn't afraid of compiling applications because I was still using Slackware Linux on my workstation. I really liked FreeBSD. In early 2004, my workstation died. When I replaced it, I decided to give FreeBSD a try as a desktop. The problem was that my previous disk was formatted using ReiserFS and there was no way to read that on FreeBSD. As I couldn't have a second computer to have Linux and FreeBSD in parallel, I had this crazy idea of adding ReiserFS support to FreeBSD. Even today, I don't understand how I could think this was the best solution to get back my data from that disk. But I eventually ported that code.

“FreeBSD is a great project to learn things. It covers many, many topics in a single big project. It is both very advanced and still has room for improvement in many areas.”

—JEAN-SÉBASTIEN

.....

How did you end up becoming a committer?

Nikolai: I started out with a few bug reports here and there. Sometime later, I took the games/wtf port under my care. It comes from the NetBSD base system and I over-engineered a process that crawls the NetBSD CVS repo and uploads changes to SourceForge. I scaled up my contribution from there, and 600-something PRs later, Matthew Seaman agreed to take me under his mentorship.

Konrad: During my bachelor's degree I had some courses about OS development. They were based on Minix and Linux, and I enjoyed them a lot.

I think I was lucky to have such good teachers. As a FreeBSD user, I wanted to contribute to this OS, but I didn't know where to start. In 2013, I decided to apply for Google Summer of Code. Having a mentor who could help you find the way to solve a problem sounded great. It was not easy for me to come up with something that I could do for FreeBSD. Helpfully, FreeBSD had a wiki page with ideas for GSoC students. I picked some of them and asked Edward Tomasz Napierała (trasz@) for his opinion. Finally I sent a proposal for Unattended Encrypted Kernel Crash Dumps. It was accepted and I was mentored by Gleb Kurtsov (gleb@). The same year, I went to Malta for EuroBSDcon as a GSoC student to give a short talk about my project. The first day, I met other FreeBSD developers, including Paweł Jakub Dawidek (pjd@) during a DevSummit. I found them extremely welcoming and helpful. A half year later, Paweł asked me if I would like to work with him on Fudo. The idea of playing with FreeBSD during my free and work time seemed great.

During the next three years, I followed the FreeBSD Project and attended FreeBSD DevSummits. Last year, I finally finished encrypted kernel crash dumps and Paweł decided to propose me for a source commit bit. The core approved it, and I could commit EKCD myself.

“

If you want to become a committer, submit high-quality PRs, be responsive to PRs against ports you maintain, and pick up unmaintained ports to maintain. ” —LARRY

Larry: I've been maintaining sysutils/lsof and mail/dovecot2-pigeonhole for a while, as well as contributing lots of PRs with bugs and issues, including a few release showstoppers. I've been in the community for a long time, and Adam Weinberger suggested I become a committer after he and I worked on the latest Dovecot version together—as we worked well together.

Jean-Sébastien: At the time I ported ReiserFS, I was working for a French telco, Cegetel. When I posted the patch to freebsd-current@, two Cegetel coworkers came to me, surprised: Maxime Henrion (mux@) and Olivier Houchard (cognet@). I was as surprised as they were because I had no idea they were FreeBSD com-

mitters! In October 2004, Scott Long (scottl@) offered me an src commit bit, and Maxime agreed to mentor me. I accepted and joined the Project in November 2004. I finished the work on read-only ReiserFS and committed it to the tree. It was first published in FreeBSD 6.0-RELEASE.

Since then, I have done a few things in FreeBSD: improvements to the Synaptics touchpad support in psm(4) and some bug fixes here and there, including the Ports tree. It's only since January 2013 that I started to contribute seriously again. My laptop had a Mobility Radeon HD 5870 GPU which was barely supported and it was getting worse with the drop of “user mode-setting” for AMD GPUs. I contacted Konstantin Belousov (kib@) and Alexander Kabaev (kan@) to see if and how I could port the Radeon kernel driver from Linux to FreeBSD. I finished that port around August 2013.

In the process, I became more involved with the graphics stack ports: mainly the X.Org server and Mesa. I worked with Koop Mast (kwm@) and Niclas Zeising (zeising@). They guided me into the Ports tree. With all those newly acquired skills, I submitted more contributions to the Ports tree—outside the graphics stack area. I even took maintainership of graphics/darktable, a digital photo development application. And I submitted many updates to the ports packaging tools from my current employer, Pivotal (tools around Cloud Foundry, Bosh, and Concourse) and created the lang/rust-nightly and devel/cargo (Rust package manager and build tool) ports. In January 2017, Baptiste Daroussin (bapt@) and Antoine Brodin (antoine@) offered to mentor me for a ports commit bit.

.....
How has your experience been since joining the FreeBSD Project? Do you have any advice for readers who may be interested in also becoming a FreeBSD committer?

Nikolai: I became more active since joining the FreeBSD Project. There is no shortage of useful work to do and my involvement became an always-present way to spend any chunk of time. Have an hour to spare? Pick up some PRs and work on them! Recognizing this, it's fun to find contributors with neglected patches and keep them hooked and on track to contribute more. The pain point of contributing to a project run by volunteers (or it has been for me) is that PRs

might sit neglected for months at a time. The trick is to not get discouraged and to find and prod people who might have spare cycles to work on these and get them committed.

Konrad: During the last four years, I had the pleasure of using or working with audit, bhyve, Capsicum, jails, pkg, ZFS, and more, and I also met a lot of nice people during six FreeBSD DevSummits. If you want to become a FreeBSD committer, do the same. Meet people, even via FreeBSD mailing lists, and run FreeBSD as your desktop to find interesting tools that you could develop yourself.

Larry: If you want to become a committer, submit high-quality PRs, be responsive to PRs against ports you maintain, and pick up unmaintained ports to maintain. I've only been a committer for three weeks as I write this, so I don't have much more experience.

Jean-Sébastien: I enjoy being part of the FreeBSD Project every day, as being able to contribute to a critical project is very rewarding.

The FOSS world taught me everything I know about computers and influenced my life outside of IT in a positive way. I'm thankful I can pay my debt today.

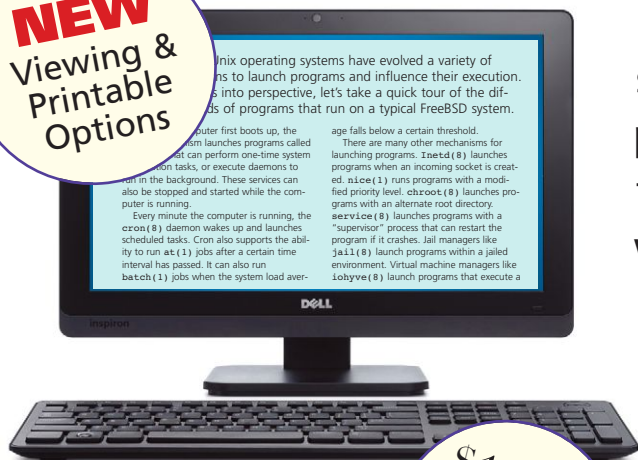
The advice I would give is to contribute to whatever area and topic that interests you. The apparent "difficulty" of the task is unimportant. There are people more than knowledgeable who will help you. Don't hesitate to discuss. You will learn whatever you need and you'll meet great people. So don't be afraid! Just go!

FreeBSD is a great project to learn things. It covers many, many topics in a single big project. It is both very advanced and still has room for improvement in many areas. If at some point you find networking is boring, you have the opportunity to work on something else. I didn't know the graphics stack, for example. •

DRU LAVIGNE is a doc committer for the FreeBSD Project and Chair of the BSD Certification Group.



NEW
Viewing &
Printable
Options



The DE, like the App, is an individual product. You will get an email notification each time an issue is released.

\$19.99
YEAR SUB
\$6.99
SINGLE COPY

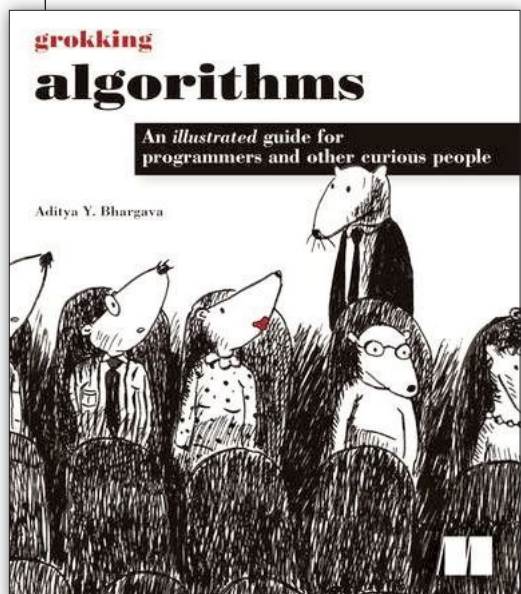
The Browser-based Edition (DE) is now available for viewing as a PDF with printable option.

The Browser-based DE format offers subscribers the same features as the App, but permits viewing the *Journal* through your favorite browser. Unlike the Apps, you can view the DE as PDF pages and print them.

To order a subscription, or get back issues, and other Foundation interests, go to

www.freebsdoundation.org

BOOKreview by Steven Kreuzer



GROKING ALGORITHMS

An illustrated guide for programmers and other curious peopleBy Aditya Y. Bhargava

Publisher.....Manning Publications (2016)

Print List Price\$44.99 (including digital)

Digital List Price\$35.99

ISBN9781617292231

Pages256

I purchased a copy of *Grokking Algorithms* on a whim based on a comment I happened to stumble upon on some random post on Hacker News. Because I don't have a computer science background, from time to time I like to challenge myself with small programming exercises, and I figured this book might have some tips and tricks that I could use down the line. When my copy finally arrived, I started to skim through it, and right in the first chapter I found a much better solution to a programming problem I had been working on. I sat down at my computer, replaced my code to search through an array with the simple binary search algorithm described in the book, and finally managed to get this small application to run fast enough that it wasn't getting killed due to excessive execution time when I submitted the code to the programming challenge website for grading. I was thrilled when I finally got a green checkmark next to each of the test cases, and I was immediately hooked on this book.

When you compare *Grokking Algorithms* to other books on the same subject, the first thing you will notice is how thin this book is when you compare it with all the others. This is by design, because the goal of this book is to be an easy on-ramp to learning more about algorithms and to help dispel the myth that algorithms are an obscure and complex subject. In the past, these types of topics were only covered in thick and heavy textbooks, and

the only way to make this material accessible was to have someone with a strong background in math and computer science help guide you through.

What I truly love about this book is that the author does a fantastic job of taking material that on the surface can seem complex and overwhelming to the casual reader, and making it extremely accessible regardless of your technical background. This is accomplished by taking real-world problems and mixing them in with plenty of illustrated explanations. The result is very unique and engaging, especially if you consider yourself to be a visual learner.

Throughout the book the example code provided for each algorithm is presented in Python, but the implementations are easy enough to translate to whatever language you would prefer to work in. Considering the popularity of the algorithms the author chose, I am sure that examples in your preferred language can be easily found. Because of that, instead of just explaining the algorithm, the author spends most chapters detailing the trade-offs you would encounter with each algorithm and focuses more on showing you techniques for problem solving. As a result, a large chunk of Chapter 1 is spent helping the reader get a better understanding about the running time of different algorithms. Big O notation has always been a little confusing to me, and so the parts of this book that provide the most value to me are the seven pages covering algorithm run-

time using very simple explanations and illustrations. The author also included a practical example on how you could visualize Big O runtimes with a few pieces of paper and a pen.

Another interesting feature of this book is that if you buy a hard copy of it, the publisher will throw in a free digital copy as well. For a variety of reasons, I still prefer physical copies of books, especially ones on technical topics. However, from time to time you might find yourself at home wanting to look up something in a book you left at work, or you just might want to give your back a break, and that is when having a digital copy comes in handy. With most other publishers, the cost of the physical and digital copies is almost the same, which makes it hard to justify buying both, and fortunately with Manning Publications, you don't have to make that tough decision. Hopefully, this is something we will see other publishers start to do as well.

If you are new to data structures and algorithms and would like to learn more or just want to experience a very fresh and unique perspective on them, I strongly encourage you to pick up a copy

“Very unique and engaging, especially if you consider yourself to be a visual learner.”



of this book. While *Grokking Algorithms* is a quick read—you can get through it in an afternoon—I have a strong feeling this will be one of those books that you will keep around in your collection and pull off the shelf for a quick refresher every once in a while. While I think I will get plenty of value out of this book as the years go by, what most excites me is anticipating when my daughter is old enough to take my dog-eared copy so that she can work through the chapters in the book, take that knowledge, and apply the same solutions to problems she might be working on. •

STEVEN KREUZER is a FreeBSD Developer and Unix Systems Administrator with an interest in retro-computing and air-cooled Volkswagens. He lives in Queens, New York, with his wife, daughter, and dog.

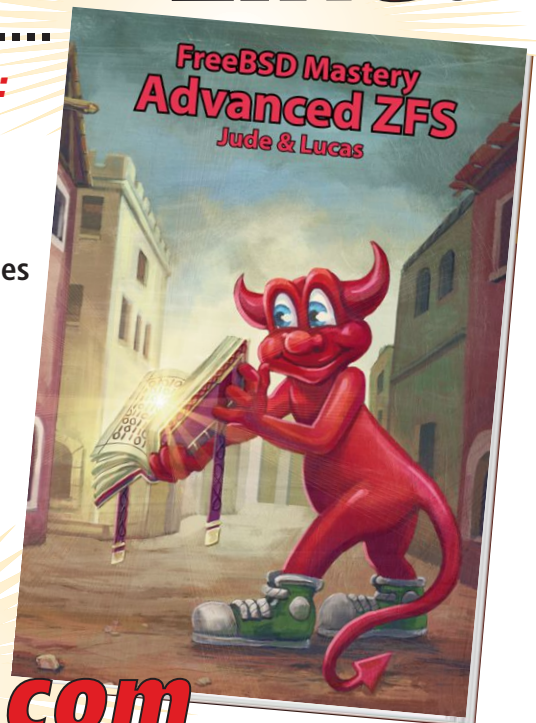
ZFS experts make their servers **ZING!**

Now you can too. Get a copy of.....

Choose ebook, print or combo. You'll learn:

- Use boot environments to make the riskiest sysadmin tasks boring.
- Delegate filesystem privileges to users.
- Containerize ZFS datasets with jails.
- Quickly and efficiently replicate data between machines
- Split layers off of mirrors.
- Optimize ZFS block storage.
- Handle large storage arrays.
- Select caching strategies to improve performance.
- Manage next-generation storage hardware.
- Identify and remove bottlenecks.
- Build screaming fast database storage.
- Dive deep into pools, metaslabs, and more!

Link to: [**http://zfsbook.com**](http://zfsbook.com)



WHETHER YOU MANAGE A SINGLE SMALL SERVER OR INTERNATIONAL DATACENTERS, SIMPLIFY YOUR STORAGE WITH **FREEBSD MASTERY: ADVANCED ZFS**. GET IT TODAY!

svnUPDATE

by Steven Kreuzer

This past year was an incredible one for the FreeBSD Project. Over 10,000 commits were made to src, the ports tree is getting close to 27,000 packages, 23 new developers were granted commit bits, and both FreeBSD 10.3 and 11 were released. For a multitude of reasons, I have a strange feeling that when we look back at 2016, the word “interesting” will be used quite a bit, and the FreeBSD Project was not immune. While activity in base tapered off a bit in December, keeping with the theme, I did see a few commits that I would describe as “interesting.”

clang, llvm, lldb, compiler-rt and libc++ have been upgraded to 3.9.0, and lld has been imported (<https://svnweb.freebsd.org/changeset/base/309124>).

The most exciting part of this commit is that it also brings lld into the base system, which is a major milestone for the llvm project. It is the first release that can link real-world large userland programs, including LLVM/Clang/LLD themselves. In fact, it can now be used to produce most userland programs distributed as part of FreeBSD.

Add WITH_LLD_AS_LD build knob (<https://svnweb.freebsd.org/changeset/base/309142>).

If set, it installs LLD as /usr/bin/ld. LLD (as of version 3.9) is not capable of linking the world and kernel, but can self-host and link many substantial applications. GNU ld continues to be used for the world and kernel build, regardless of how this knob is set. It is on by default for arm64, and off for all other CPU architectures.

Use buffer pager for NFS (<https://svnweb.freebsd.org/changeset/base/308980>).

The pager, due to its construction, implements clustering for the page-ins. In particular, buildworld load demonstrates reduction of the READ RPCs from 39k down to 24k. No change in real or CPU time was observed.

Support for Ingenic XBurst JZ4780 and X1000 systems on chips (<https://svnweb.freebsd.org/changeset/base/308857>).

Ingenic Semiconductor designs CPU Microarchitectures based around the MIPS instruction set. This change introduces support for the Imgtec CI20 and Ingenic CANNA single board computers.

Capsicumize some trivial stdio programs (<https://svnweb.freebsd.org/changeset/base/308432>).

We are starting to see more and more applications take advantage of Capsicum, a lightweight OS capability and sandbox framework. A recent commit added capsicum support to the echo, sleep, basename, dc, dirname, fold, getopt, locate, logname, printenv, and yes, since these userland utilities only interact with stdio.

Performance improvements for pw operations that edit /etc/group or /etc/passwd (<https://svnweb.freebsd.org/changeset/base/308806>).

285050 fixed a bug in pw that could lead to /etc/passwd or /etc/group corruption on power loss. However, it was fixed by opening those files with O_SYNC, which is very slow, especially on ZFS. This change replaces O_SYNC with

appropriately placed `fsync()`s instead, which is much faster. Using a ZFS `tmpdir`, the time to run `pw`'s `kyua` tests drops from 245s to 35s.

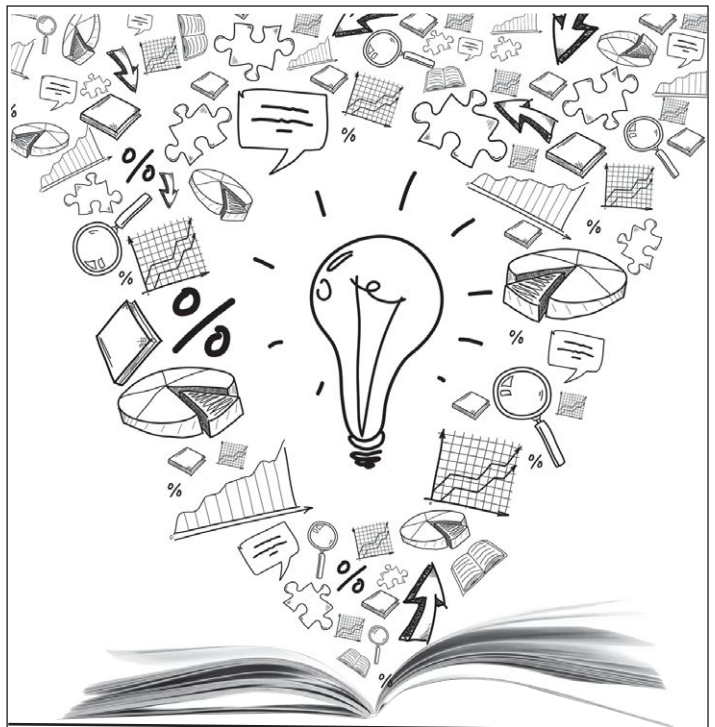
Concurrency Kit is now part of the base system (<https://svnweb.freebsd.org/changeset/base/309266>).

CK is a BSD-licensed toolkit providing concurrency primitives, safe memory reclamation mechanisms, and non-blocking data structures for the research, design, and implementation of high performance concurrent systems.

Additional updates in contrib/

- Subversion has been updated to version 1.9.5 (<https://svnweb.freebsd.org/changeset/base/309356>)
- ntp has been upgraded to version 4.2.8p9 (<https://svnweb.freebsd.org/changeset/base/308957>)
- ACPICA has been upgraded to 20161117 (<https://svnweb.freebsd.org/changeset/base/308953>)
- am-utils has been updated to version 6.2 (<https://svnweb.freebsd.org/changeset/base/308493>)
- jemalloc has been updated to version 4.3.1 (<https://svnweb.freebsd.org/changeset/base/308473>)
- file has been updated to version 5.29 (<https://svnweb.freebsd.org/changeset/base/308420>)
- tzdata has been updated to version 2016i (<https://svnweb.freebsd.org/changeset/base/308270>)
- libarchive has been updated to 3.2.2 (<https://svnweb.freebsd.org/changeset/base/307861>)

STEVEN KREUZER is a FreeBSD Developer and Unix Systems Administrator with an interest in retro-computing and air-cooled Volkswagens. He lives in Queens, New York, with his wife, daughter, and dog.



freeBSD JOURNAL

WRITE FOR US!

Contact Jim Maurer
(jmaurer@freebsdjournal.com)
with your article ideas.





THROUGH MARCH 2017

BY DRU LAVIGNE

Events Calendar

The following BSD-related conferences will take place in March 2017.

**SCALE
15x**



SCALE • Mar 2–5 • Pasadena, CA

<https://www.socallinuxexpo.org/scale/15x/> • The 15th annual Southern California Linux Expo will once again provide several FreeBSD-related presentations and a FreeBSD booth in the expo area. This event requires registration at a nominal fee.

AsiaBSDCon • Mar 9–12 • Tokyo, Japan

<http://2017.asiabsdcon.org/> • This is the annual BSD technical conference for users and developers on BSD-based systems. It provides several days of workshops, presentations, a Developer Summit, and an opportunity to take the BSDA certification exam in either English or Japanese. Registration is required for this event.



Thank you!

The FreeBSD Foundation would like to acknowledge the following companies for their continued support of the Project. Because of generous donations such as these we are able to continue moving the Project forward.



Are you a fan of FreeBSD? Help us give back to the Project and donate today! freebsdfoundation.org/donate/

Please check out the full list of generous community investors at freebsdfoundation.org/donate/sponsors

Uranium

**The Koum Family
Anonymous**

Iridium



Platinum



Gold



facebook **NETFLIX**

Silver

